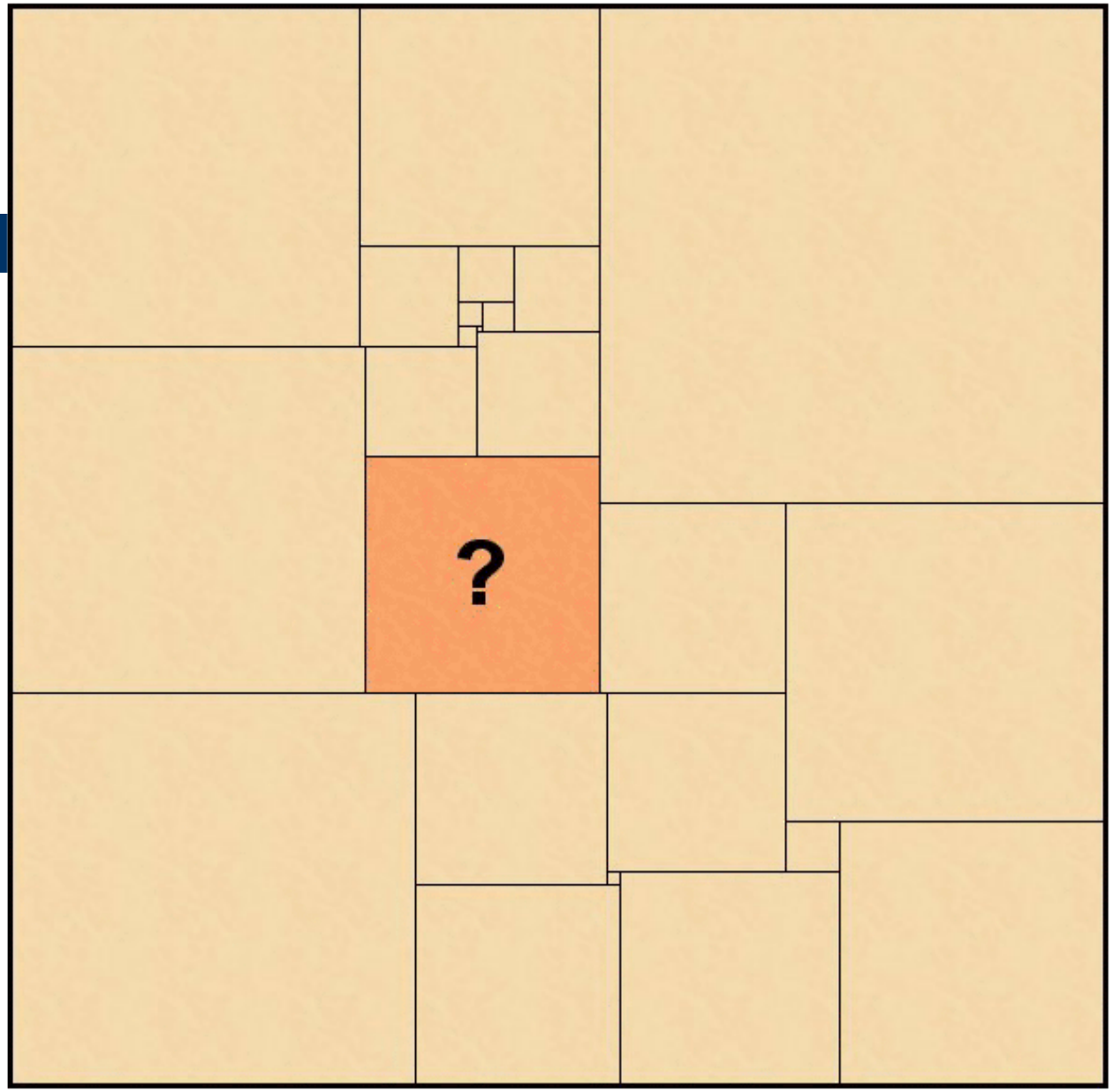
Artificial Intelligence
15. Constraint Satisfaction Problems

Course V231

Department of Computing
Imperial College, London

Jeremy Gow

The square below contains 24 smaller squares, each with a different integral size. Determine the length of the shaded square



The High IQ Exam

- From the High-IQ society entrance exam
  - Published in the Observer newspaper
  - Never been solved...
- Solved using a **constraint solver**
  - 45 minutes to **specify** as a CSP (Simon)
  - 1/100 second to solve (Sicstus Prolog)
- See the notes
  - the program, the results and the answer

Constraint Satisfaction Problems

- Set of variables $X = \{x_1, x_2, \ldots, x_n\}$
  - D**omain** for each variable (finite set of values)
  - Set of constraints
    - Restrict the values that variables can take together

- A solution to a CSP...
  - An assignment of a domain value to each variable
  - Such that no constraints are broken

Example: High-IQ problem CSP

- Variables:
  - 25 lengths (Big square made of 24 small squares)
- Values:
  - Let the tiny square be of length 1
  - Others range up to about 200 (at a guess)
- Constraints: lengths have to add up
  - e.g. along the top row
- Solution: set of lengths for the squares
- Answer: length of the 17th largest square

What we want from CSP solvers

- One solution
  - Take the first answer produced
- The ‘best’ solution
  - Based on some measure of optimality
- All the solutions
  - So we can choose one, or look at them all
- That no solutions exist
  - Existence problems (common in mathematics)

Formal Definition of a Constraint

- Informally, relationships between variables
  - e.g., $x \neq y$, $x > y$, $x + y < z$
- Formal definition:
  - Constraint $C_{xyz...}$ between variables x, y, z, …
  - $C_{xyz...} \subseteq D_x \times D_y \times D_z \times$ ... (a subset of all tuples)
- Constraints are a set of which tuples **ARE** allowed in a solution
- Theoretical definition, not implemented like this

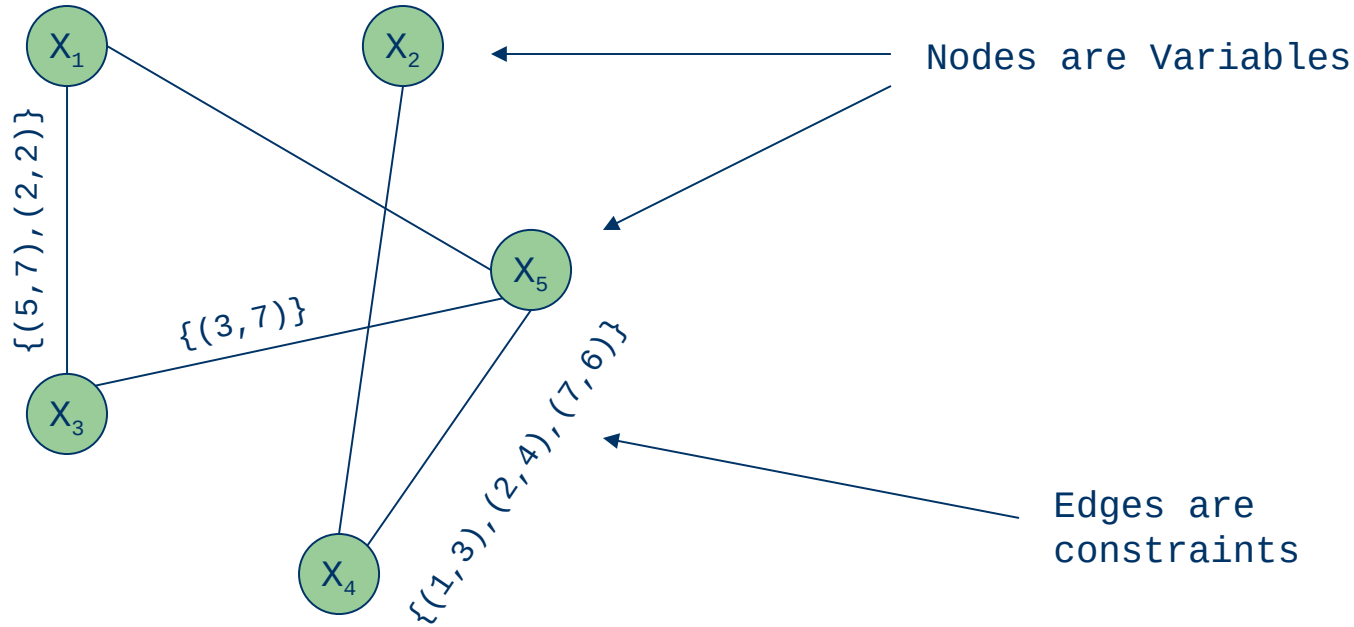
Example Constraints

- Suppose we have two variables: x and y
- x can take values {1,2,3}
- y can take values {2,3}
- Then the constraint x=y is written:
  - {(2,2), (3,3)}
- The constraint x < y is written:
  - {(1,2), (1,3), (2,3)}

Binary Constraints

- Unary constraints: involve only one variable
  - Preprocess: re-write domain, remove constraint
- Binary constraints: involve two variables
  - Binary CSPs: all constraints are binary
  - Much researched
    - All CSPs can be written as binary CSPs (no details here)
    - Nice graphical and Matrix representations
  - Representative of CSPs in general

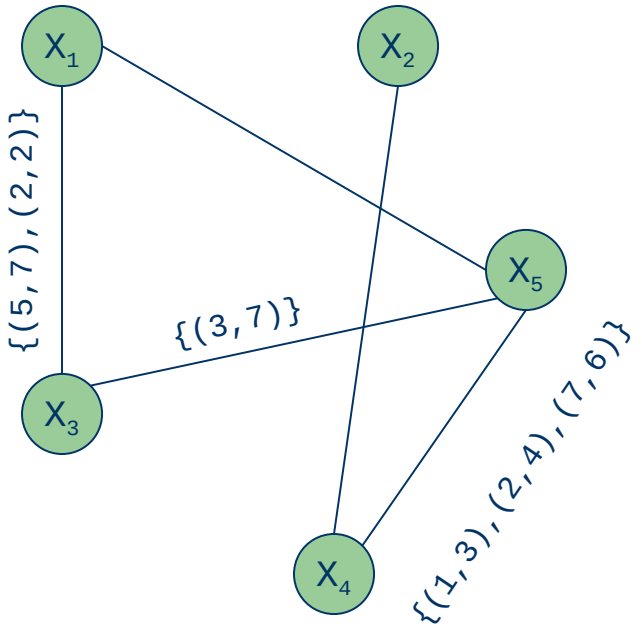
Binary Constraint Graph



Matrix Representation for Binary Constraints

C	1	2	3	4	5	6	7
1			X				
2				X			
3							
4							
5							
6							
7						X	

Matrix for the constraint between X_4 and X_5 in the graph



Random Generation of CSPs

- Generation of random binary CSPs
  - Choose a number of variables
  - Randomly generate a matrix for every pair of variables
- Used for benchmarking
  - e.g. efficiency of different CSP solving techniques
- Real world problems often have more **structure**

Rest of This Lecture

- **Preprocessing**: arc consistency
- **Search**: backtracking, forward checking
- **Heuristics**: variable & value ordering
- Applications & advanced topics in CSP

Arc Consistency

- In binary CSPs
  - Call the pair (x, y) an **arc**
  - Arcs are ordered, so (x, y) is not the same as (y, x)
  - Each arc will have a single associated constraint C_{xy}
- An arc (x,y) is consistent if
  - For all values *a* in D_x, there is a value *b* in D_y
    - Such that the assignment x = *a*, y = *b* satisfies C_{xy}
  - Does not mean (y, x) is consistent
  - Removes zero rows/columns from C_{xy}'s matrix

Making a CSP Arc Consistent

- To make an arc (x,y) consistent
  - remove values from D_x which make it inconsistent
- Use as a **preprocessing** step
  - Do this for every arc in turn
  - Before a search for a solution is undertaken
- Won't affect the solution
  - Because removed values would break a constraint
- Does not remove all inconsistency
  - Still need to search for a solution

Arc Consistency Example

- Four tasks to complete (A,B,C,D)
- Subject to the following constraints:

Task	Duration	Precedes
A	3	B,C
B	2	D
C	4	D
D	2	

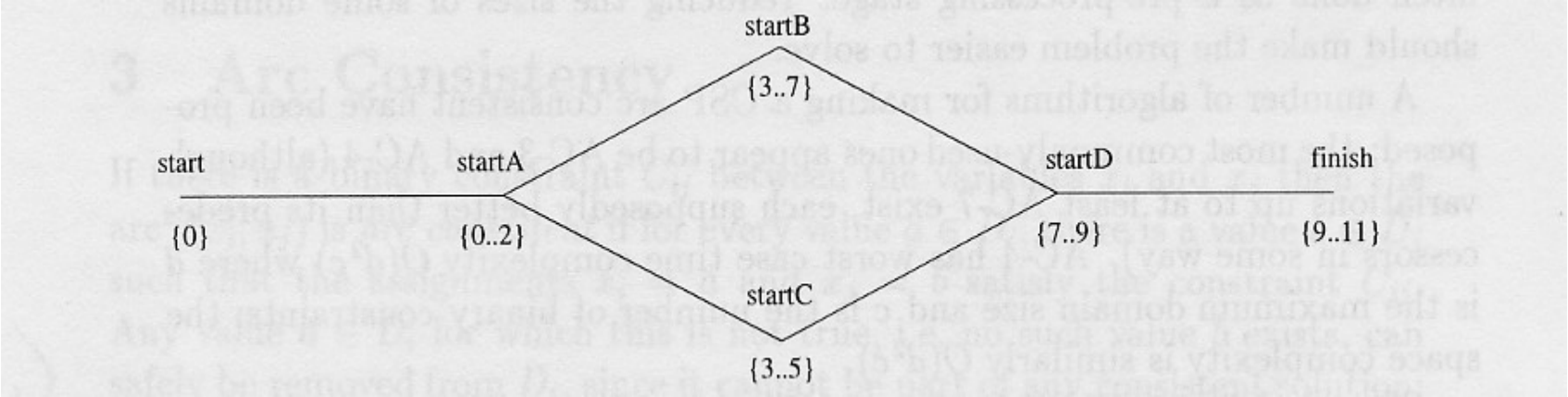
C1	start ≤ startA
C2	startA + 3 ≤ startB
C3	startA + 3 ≤ startC
C4	startB + 2 ≤ startD
C5	startC + 4 ≤ startD
C6	startD + 2 ≤ finish

- Formulate this with variables for the start times
  - And a variable for the global *start* and *finish*
    - Values for each variable are {0,1,…,11}, except *start* = 0
  - Constraints: startX + durationX ≤ startY

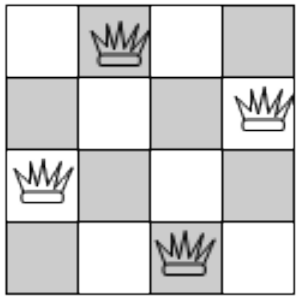
Arc Consistency Example

- Constraint C1 = {(0,0), (0,1), (0,2), …,(0,11)}
  - So, arc (start,startA) is arc consistent
- C2 = {(0,3),(0,4),…(0,11),(1,4),…,(8,11)}
  - These values for startA never occur: {9,10,11}
    - So we can remove them from D_{startA}
  - These values for startB never occur: {0,1,2}
    - So we can remove them from D_{startB}
- For CSPs with precedence constraints only
  - Arc consistency removes all values which can't appear in a solution (if you work backwards from last tasks to the first)
- In general, arc consistency is effective, but not enough

Arc Consistency Example



N-queens Example (N = 4)

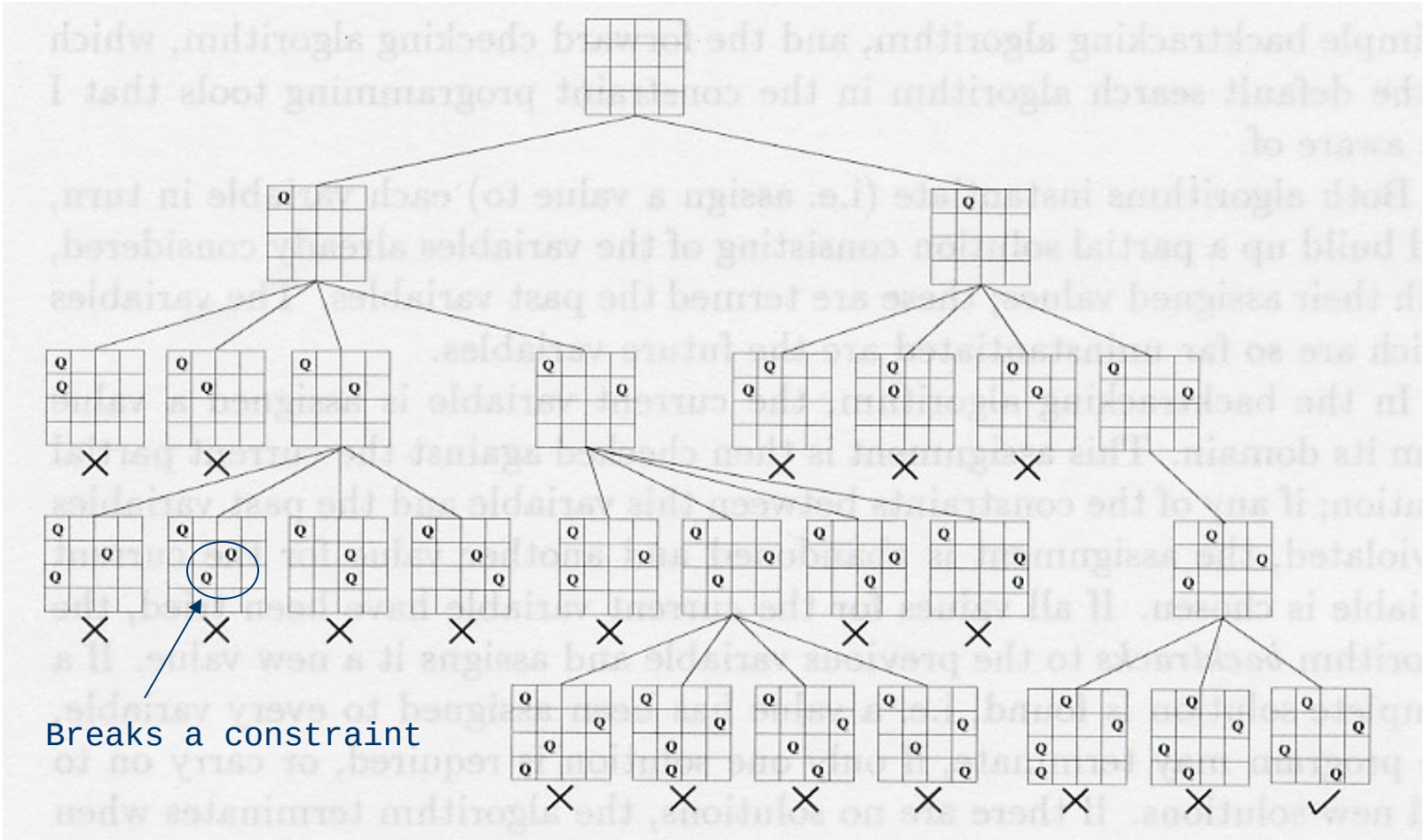


- Standard test case in CSP research
- Variables are the rows
- Values are the columns
- Constraints:
  - $C_{1,2}$ = {(1,3),(1,4),(2,4),(3,1),(4,1),(4,2)}
  - $C_{1,3}$ = {(1,2),(1,4),(2,1),(2,3),(3,2),(3,4),
    (4,1),(4,3)}
  - Etc.
  - Question: What do these constraints mean?

Backtracking Search

- Keep trying all variables in a depth first way
  - Attempt to instantiate the current variable
    - With each value from its domain
  - Move on to the next variable
    - When you have an assignment for the current variable which doesn't break any constraints
  - Move back to the previous (past) variable
    - When you cannot find any assignment for the current variable which doesn't break any constraints
    - i.e., **backtrack** when a deadend is reached

Backtracking in 4-queens

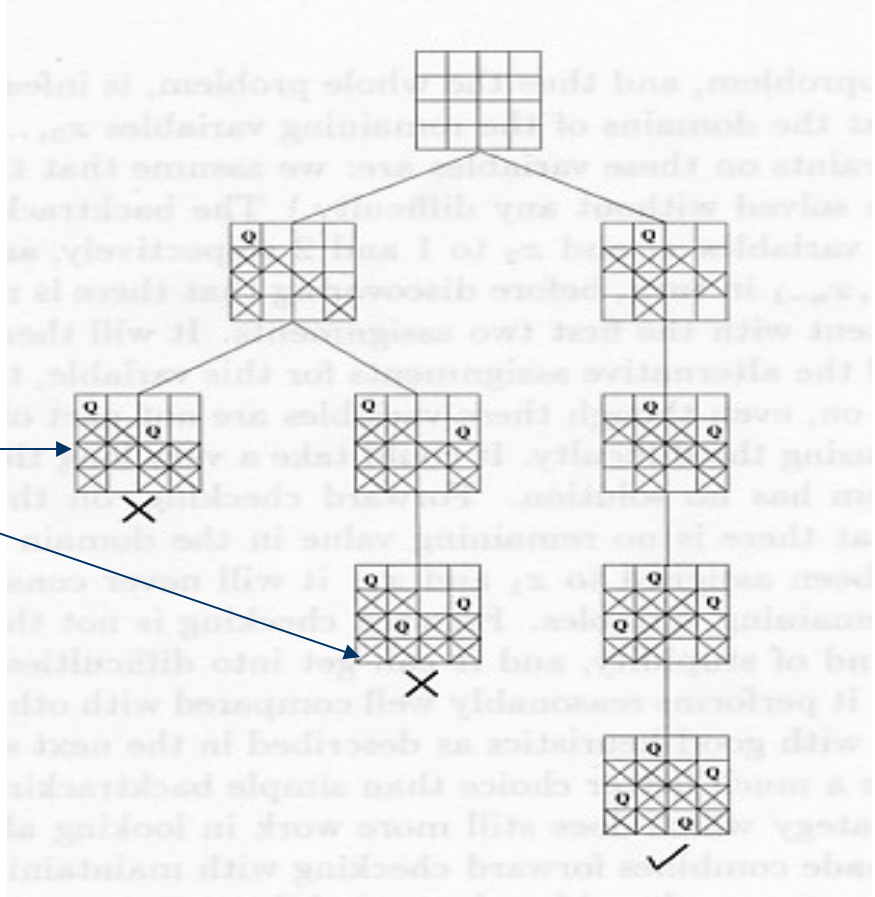


Forward Checking Search

- Same as backtracking
  - But it also looks at the future variables
    - i.e., those which haven't been assigned yet
- Whenever an assignment of value V_c to the current variable is attempted:
  - For all future variables x_f, remove (temporarily)
    - any values in D_f which, along with V_c, break a constraint
  - If x_f ends up with no variables in its domain
    - Then the current value V_c must be a "no good"
    - So, move onto next value or backtrack

Forward Checking in 4-queens

Shows that this assignment is no good



Heuristic Search Methods

- Two choices made at each stage of search
  - Which order to try to instantiate variables?
  - Which order to try values for the instantiation?
- Can do this
  - Statically (fix before the search)
  - Dynamically (choose during search)

- May incur extra cost that makes this ineffective

Fail-First Variable Ordering

- For each future variable
  - Find size of domain after forward checking pruning
  - Choose the variable with the fewest values left
- Idea:
  - We will work out quicker that this is a dead-end
  - Because we only have to try a small number of vars.
  - “Fail-first” should possibly be “dead end quickly”
- Uses information from forward checking search
  - No extra cost if we’re already using FC

Dynamic Value Ordering

- Choose value which most likely to succeed
  - If this is a dead-end we will end up trying all values anyway
- Forward check for each value
  - Choose one which reduces other domains the least
  - ‘Least constraining value’ heuristic
- Extra cost to do this
  - Expensive for random instances
  - Effective in some cases

Some Constraint Solvers

- Standalone solvers
  - ILOG (C++, popular commercial software)
  - JaCoP (Java)
  - Minion (C++)
  - ...
- Within Logic Programming languages
  - Sicstus Prolog
  - SWI Prolog
  - ECLiPSe
  - ...

Overview of Applications

- Big advantages of CSPs are:
  - They cover a big range of problem types
  - It is usually easy to come up with a quick CSP spec.
  - And there are some pretty fast solvers out there
  - Hence people use them for quick solutions
- However, for seriously difficult problems
  - Often better to write bespoke CSP methods
  - Operational research (OR) methods often better

Some Mathematical Applications

- Existence of algebras of certain sizes
  - QG-quasigroups by John Slaney
  - Showed that none exists for certain sizes
- Golomb rulers
  - Take a ruler and put marks on it at integer places such that *no pair* of marks have the same length between them.
    - Thus, the all-different constraint comes in
  - Question: Given a particular number of marks
    - What's the *smallest* Golomb ruler which accommodates them

Some Commercial Applications

- Sports scheduling
  - Given a set of teams
    - All who have to play each other twice (home and away)
    - And a bunch of other constraints
  - What is the best way of scheduling the fixtures
  - Lots of money in this one
- Packing problems
  - E.g., How to load up ships with cargo
    - Given space, size and time constraints

Some Advanced Topics

- Formulation of CSPs
  - It's very easy to specify CSPs (this is an attraction)
    - But some are worse than others
  - There are *many* different ways to specify a CSP
  - It's a highly skilled job to work out the best
- Automated reformulation of CSPs
  - Given a simple formulation
    - Can an agent change that formulation for the better?
    - Mostly: what choice of variables are specified
    - Also: automated discovery of additional constraints
      - Can we add in extra constraints the user has missed?

More Advanced Topics

- Symmetry detection
  - Can we spot whole branches of the search space
    - Which are exactly the same (symmetrical with) a branch we have already (or are going to) search
  - Humans are good at this
  - Can we get search strategies to do this automatically?
- Dynamic CSPs
  - Solving of problems which change while you are trying to solve them
  - E.g. a new package arrives to be fitted in